

Advice Complexity of the Online Induced Subgraph Problem*

Dennis Komm¹, Rastislav Kráľovič², Richard Kráľovič³, and Christian Kudahl⁴

- 1 Dept. of Computer Science, ETH Zurich; dennis.komm@inf.ethz.ch
- 2 Dept. of Computer Science, Comenius University; kralovic@dcs.fmph.uniba.sk
- 3 Google Inc., Switzerland; richard.kralovic@dcs.fmph.uniba.sk
- 4 Dept. of Mathematics and Computer Science, University of Southern Denmark
kudahl@imada.sdu.dk

Abstract

Several well-studied graph problems aim to select a largest (or smallest) induced subgraph with a given property of the input graph. Examples of such problems include maximum independent set, maximum planar graph, maximum induced clique, maximum acyclic subgraph (a.k.a. minimum feedback vertex set), and many others. In online versions of these problems, vertices of the graph are presented in an adversarial order, and with each vertex, the online algorithm must irreversibly decide whether to include it into the constructed subgraph, based only on the subgraph induced by the vertices presented so far. We study the properties that are common to all these problems by investigating the generalized problem: for an arbitrary but fixed hereditary property π , find some maximal induced subgraph having π . We study this problem from the point of view of advice complexity, i. e., we ask how some additional information about the yet unrevealed parts of the input can influence the solution quality. We evaluate the information in a quantitative way by considering the best possible advice of given size that describes the unknown input. Using a result from Boyar et al. [STACS 2015, LIPIcs 30], we give a tight trade-off relationship stating that for inputs of length n roughly n/c bits of advice are both needed and sufficient to obtain a solution with competitive ratio c , regardless of the choice of π , for any c (possibly a function of n). This complements the results from Bartal et al. [SIAM Journal on Computing 36(2), 2006] stating that, without any advice, even a randomized algorithm cannot achieve a competitive ratio better than $\Omega(n^{1-\log_4 3 - o(1)})$.

Surprisingly, a similar result cannot be obtained for the symmetric problem: for a given cohereditary property π , find a minimum subgraph having π . We show that the advice complexity of this problem varies significantly with the choice of π .

We also consider the so-called preemptive online model, inspired by some application mainly in networking and scheduling, where the decision of the algorithm is not completely irreversible. In particular, the algorithm may discard some vertices previously assigned to the constructed set, but discarded vertices cannot be reinserted into the set again. We show that, for the maximum induced subgraph problem, preemption cannot help much, giving a lower bound of $\Omega(n/(c^2 \log c))$ bits of advice needed to obtain competitive ratio c , where c is any increasing function bounded by $\sqrt{n/\log n}$. We also give a linear lower bound for c close to 1.

1 Introduction

Online algorithms get their input gradually, and this way have to produce parts of the output without full knowledge of the instance at hand, which is a large disadvantage compared to

* Supported in part by the Villum Foundation and the Stibo-Foundation and SNF grant 200021-146372.



licensed under Creative Commons License CC-BY



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

classical *offline computation*, yet a realistic model of many real-world scenarios [5]. Most of the offline problems have their online counterpart. Instead of asking about the time and space complexity of algorithms to solve a computational problem, *competitive analysis* is commonly used as a tool to study how well online algorithms perform [5, 17] without any time or space restrictions; the analogous offline measurement is the analysis of the *approximation ratio*. A large class of computational problems for both online and offline computation are formulated on graphs; we call such problems (online) *graph problems*.

In this paper, we deal with problems on unweighted undirected graphs that are given to an online algorithm vertex by vertex in consecutive discrete time steps. Formally, we are given a graph $G = (V, E)$, where $|V| = n$, with an ordering \prec on V . Without loss of generality, assume $V = \{v_1, \dots, v_n\}$, and $v_1 \prec v_2 \prec \dots \prec v_n$ specifies the order in which the vertices of G are presented to an online algorithm; this way, the vertex v_i is given in the i th time step. Together with v_i , all edges $\{v_j, v_i\} \in E$ are revealed for all $v_j \prec v_i$. If v_i is revealed, an online algorithm must decide whether to accept v_i or discard it. Neither G nor n are known to the online algorithm. We study two versions of online problems; with and without preemption. In the former case, the decision whether v_i is accepted or not is definite. In the latter case, in every time step, the online algorithm may preempt (discard) some of the vertices it previously accepted; however, a vertex that was once discarded cannot be part of the solution anymore.

For an instance $I = (v_1, \dots, v_n)$ of some graph problem, we denote by $\text{ALG}(I)$ the solution computed by some online algorithm ALG ; $\text{OPT}(I)$ denotes an optimal solution for I , which can generally only be computed with the full knowledge of I . We assume that I is constructed in an *adversarial manner* to give worst-case bounds on the solution quality of any online algorithm. This means that we explicitly think of I as being given by an adversary that knows ALG and wants to make it perform as poorly as possible; for more details, we refer to the standard literature [5].

For maximization problems with an associated *profit function* profit , an online algorithm ALG is called *c-competitive* if, for every instance I of the given problem, it holds that

$$\text{profit}(\text{ALG}(I)) \geq 1/c \cdot \text{profit}(\text{OPT}(I)) ; \quad (1)$$

likewise, for minimization problems with a *cost function* cost , we require

$$\text{cost}(\text{ALG}(I)) \leq c \cdot \text{cost}(\text{OPT}(I)) \quad (2)$$

for every instance I . In this context, c may be a positive constant or a function that increases with the input length n . We will use c and $c(n)$ interchangeably to refer to the competitive ratio; the latter is simply used to emphasize that c may depend on n .

Throughout this paper, \log denotes the binary logarithm \log_2 . For $q \in \mathbb{N}$, let $[q] = \{0, 1, \dots, q-1\}$.

Instead of studying specific graph problems, in this paper, we investigate a large class of such problems, which are defined by *hereditary properties*. This class includes many well-known problems such as *maximum independent set*, *maximum planar graph*, *maximum induced clique*, and *maximum acyclic subgraph*. We call any collection of graphs a *graph property* π . A graph has property π if it is in the collection. We only consider properties that are *non-trivial*, i. e., they are both true for infinitely many graphs and false for infinitely many graphs. A property is called *hereditary* if it holds that, if a graph G satisfies π , then also any induced subgraph G' of G satisfies π ; conversely, it is called *cohereditary* if it holds that, if a graph G satisfies π , and G is an induced subgraph of G' , then also G' satisfies π . For a graph $G = (V, E)$ and a subset of vertices $S = \{v_1, \dots, v_i\} \subseteq V$, let $G[S]$ (or $G[v_1, \dots, v_i]$) denote

the subgraph of G induced by the vertices from S . For a graph $G = (V, E)$, let $\overline{G} = (V, \overline{E})$ be the complement of G , i. e., $\{u, v\} \in \overline{E}$ if and only if $\{u, v\} \notin E$. Let K_n denote the complete graph on n vertices, and let \overline{K}_n denote the independent set on n vertices. We consider the online version of the problem of finding maximal (minimal, respectively) induced subgraphs satisfying a hereditary (cohereditary, respectively) property π , denoted by $\text{MAX-}\pi$ ($\text{MIN-}\pi$, respectively). For the ease of presentation, we will call such problems *hereditary* (*cohereditary*, respectively) *problems*. Let $S_{\text{ALG}} := \text{ALG}(I)$ denote the set of vertices accepted by some online algorithm ALG for some instance I of a hereditary problem. Then, for $\text{MAX-}\pi$, the profit of ALG is $|S_{\text{ALG}}| := \text{profit}(\text{ALG}(I))$ if $G[S_{\text{ALG}}]$ has the property π and $-\infty$ otherwise; the goal is to maximize the profit. Conversely, for $\text{MIN-}\pi$, the cost of ALG is $|S_{\text{ALG}}| := \text{cost}(\text{ALG}(I))$ if $G[S_{\text{ALG}}]$ has the property π and ∞ otherwise; the goal is to minimize the cost. As an example, consider the *online maximum independent set* problem; the set of all independent sets is clearly a hereditary property (every independent set is a feasible solution, and every induced subset of an independent set is again an independent set). With every vertex revealed, an online algorithm needs to decide whether it becomes part of the solution or not. The goal is to compute an independent set that is as large as possible; the profit of the solution thus equals $|S_{\text{ALG}}|$. It is straightforward to define the problem without or with preemption.

In this paper, we study *online algorithms with advice* for hereditary and cohereditary problems. In this setup, an online algorithm is equipped with an additional resource that contains information about the instance it is dealing with. A related model was originally introduced by Dobrev et al. [8]. Revised versions were introduced by Emek et al. [10], Böckenhauer et al. [4], and Hromkovič et al. [11]. Here, we use the model of the latter two papers. Consider an input $I = (v_1, \dots, v_n)$ of a hereditary problem. An *online algorithm with advice* computes the output sequence $\text{ALG}^\phi(I) = (y_1, \dots, y_n)$ such that y_i is computed from ϕ, v_1, \dots, v_i , where ϕ is the content of the advice tape, i. e., an infinite binary sequence. We denote the cost (profit, respectively) of the computed output by $\text{cost}(\text{ALG}^\phi(I))$ ($\text{profit}(\text{ALG}^\phi(I))$, respectively). The algorithm ALG is *c-competitive with advice complexity* $b(n)$, if for every n and for each I of length at most n , there exists some ϕ such that $\text{cost}(\text{ALG}^\phi(I)) \leq c \cdot \text{cost}(\text{OPT}(I))$ ($\text{profit}(\text{ALG}^\phi(I)) \geq (1/c) \cdot \text{profit}(\text{OPT}(I))$, respectively) and at most the first $b(n)$ bits of ϕ have been accessed by ALG .¹

The motivation for online algorithms with advice is mostly of theoretical nature, as we may think of the information necessary and sufficient to compute an optimal solution as the *information content* of the given problem [11]. Moreover, there is a non-trivial connection to randomized online algorithms [3, 12]. Lower bounds from advice complexity often translate to lower bounds for semi-online algorithms. One could consider if knowing some small parameter for an online problem (such as the length of the input or the number of requests of a certain type) could result in a much better competitive ratio. Lower bound results using advice can often help answer this question. Similarly, lookahead can be seen as a special kind of advice that is supplied to an algorithm. This way, online algorithms with advice generalize a number of concepts introduced to give online algorithms more power. However, the main question posed is, how much could any kind of (computable) information help; and maybe even more importantly, which amount of information will never help to overcome some certain threshold, no matter what this information actually is.

¹ Note that usually an additive constant is included in the definition of c -competitiveness, i. e., in (1) and (2). However, for the problems we consider, this changes the advice complexity by at most $O(\log n)$; see Remark 9 in Boyar et al. [7].

Organization, Related Work, and Results

We are mainly concerned with proving lower bounds of the form that a particular number of advice bits is necessary in order to obtain some certain output quality for a given hereditary property. We make heavy use of online reductions between generic problems and the studied ones that allow to bound the number of advice bits necessary from below. Emek et al. [10] used this technique in order to prove lower bounds for the k -server problem. The foundations of the reductions as we perform them here are due to Böckenhauer et al. [2], who introduced the *string guessing problem*, and Boyar et al. [7], who studied a problem called *asymmetric string guessing*. Mikkelsen [19] introduced a problem, which we call the *anti-string guessing problem*, and which is a variant of string guessing with a more “friendly” cost function. Our reductions rely on some results from Bartal et al. [1] that characterize hereditary properties by forbidden subgraphs together with some insights from Ramsey theory.

In Section 2, we recall some basic results from Ramsey theory, and define the generic online problems that we use as a basis of our reductions. In Section 3, we study both $\text{MAX-}\pi$ and $\text{MIN-}\pi$ in the case that no preemption is allowed; using a reduction from the asymmetric string guessing problem, we show that any c -competitive online algorithm for $\text{MAX-}\pi$ needs roughly n/c advice bits, and this is essentially tight. This complements the results from Bartal et al. [1] stating that, without any advice, even a randomized algorithm cannot achieve a competitive ratio better than $\Omega(n^{1-\log_4 3 - o(1)})$. Note that the advice complexity of the maximum independent set problem on bipartite and sparse graphs was studied by Dobrev et al. [9]. In the subsequent sections, we allow the online algorithm to use preemption. First, in Section 4, we use a reduction from the string guessing problem to show a lower bound of $\Omega(n/(c^2 \log c))$ bits of advice that are needed to obtain competitive ratio c , where c is any increasing function bounded by $\sqrt{n/\log n}$. In Section 5, using a reduction from the anti-string guessing problem, we also give a linear lower bound for c close to 1.

2 Preliminaries

Hereditary properties can be characterized by forbidden induced subgraphs as follows: if a graph G does not satisfy a hereditary property π , then any graph H such that G is an induced subgraph of H does not satisfy π neither. Hence, there is a (potentially infinite) set of minimal forbidden graphs (w.r.t. being induced subgraph) S_π such that G satisfies π if and only if no graphs from S_π are induced subgraphs of G . Conversely, any set of graphs S defines a hereditary property π_S of not having a graph from S as induced subgraph.

Furthermore, there is the following bijection between hereditary and cohereditary properties: for a hereditary property π we can define a property $\bar{\pi}$ such that a graph G satisfies $\bar{\pi}$ if and only if it does not satisfy π (it is easy to see that $\bar{\pi}$ is cohereditary), and vice versa. Hence, a cohereditary property $\bar{\pi}$ can be characterized by a set of minimal (w.r.t. being induced subgraph) *obligatory* subgraphs $S_{\bar{\pi}}$ such that a graph G has the property $\bar{\pi}$ if and only if at least one graph from $S_{\bar{\pi}}$ is an induced subgraph of G .

To each property π we can define the complementary property π^c such that a graph G satisfies π^c if and only if the complement of G satisfies π . Clearly, if π is (co)hereditary, so is π^c . Moreover, if H is forbidden (obligatory, respectively) for π , \bar{H} is forbidden (obligatory, respectively) for π^c . The following statement is due to Lund and Yannakakis.

► **Lemma 1** (Lund and Yannakakis [14], proof of Theorem 4). *Every non-trivial hereditary property π is satisfied either by all cliques or by all independent sets.*

Proof. Assume, for the sake of contradiction, that there is a hereditary property π , and

two numbers m, n , such that K_m and \overline{K}_n do not satisfy π . Let $r(m, n)$ be the Ramsey number [16], such that every graph with at least $r(m, n)$ vertices contains K_m or \overline{K}_n as induced subgraph. Since π is non-trivial, there is a graph G with more than $r(m, n)$ vertices that satisfies π . G contains either K_m or \overline{K}_n as induced subgraph, and since π is hereditary, either K_m or \overline{K}_n satisfies π . ◀

Bartal et al. proved the following theorem.

► **Theorem 2** (Bartal et al. [1] and references therein). *In the known supergraph model, any randomized algorithm for the MAX- π problem has competitive ratio $\Omega(n^{1-\log_4 3-o(1)})$, even if preemption is allowed.* ◀

The theorem is formulated in the known supergraph model, where a graph $G = (V, E)$ with n vertices is a-priori known to the algorithm, and the input is a sequence of vertices v_1, \dots, v_k . The task is to select in an online manner the subgraph of the induced graph $G[v_1, \dots, v_k]$, having property π . Note that n in the previous theorem thus refers to the size of the known supergraph, and not to the length of the input sequence. However, in the proof a graph with $n = 4^i$ vertices is considered, from which subgraphs of size 3^i are presented. Each of these instances has an optimal solution of size at least 2^i , and it is shown that any deterministic algorithm can have a profit of at most $\alpha(3/2)^i \log n$ on average, for some constant α . From that, using Yao's principle [18] as stated in [6], the result follows. The same set of instances thus yields the following result.

► **Theorem 3** (Bartal et al. [1]). *Any randomized algorithm for the MAX- π problem has competitive ratio $\Omega(n^{2/\log 3-1-o(1)})$, even if preemption is allowed.* ◀

Next, we describe some specific online problems that allow us to give lower bounds on the advice complexity using a special kind of reduction.

Böckenhauer et al. [2] introduced a very generic online problem called *string guessing with known history over alphabets of size σ* (σ -SGKH). The input is a sequence of requests (x_0, \dots, x_n) where $x_0 = n$ and for $i \geq 1$, $x_i \in \{1, \dots, \sigma\}$. The algorithm has to produce a sequence of answers $(y_1, \dots, y_n, y_{n+1})$, where $y_i \in \{1, \dots, \sigma\}$ and $y_{n+1} = \perp$ and where y_i is allowed to depend on x_0, \dots, x_{i-1} (and of course any advice bits the algorithm reads). The cost is the number of positions i for which $y_i \neq x_i$.

► **Theorem 4** (Böckenhauer et al. [2]). *Any online algorithm with advice for σ -SGKH that guesses γn bits of the input correctly must read at least*

$$\left(1 + (1 - \gamma) \log_\sigma \left(\frac{1 - \gamma}{\sigma - 1}\right) + \gamma \log_\sigma \gamma\right) n \log_2 \sigma$$

advice bits. ◀

Mikkelsen [19] introduced the problem *anti-string guessing with known history over alphabets of size σ* (anti- σ -SGKH). It is defined exactly as σ -SGKH except that the cost is the number of positions i for which $y_i = x_i$.

► **Theorem 5** (Mikkelsen [19, Theorem 11]). *Let $\sigma \geq 2$ and let $1 \leq c < \frac{\sigma}{\sigma-1}$. A c -competitive anti- σ -SGKH algorithm must read at least*

$$b \geq (1 - h_\sigma(1/c)) n \log_2 \sigma$$

bits of advice, where n is the input length. This holds even if n is known in advance. Here, h_σ is the σ -ary entropy function given by $h_\sigma(x) = x \log_\sigma(\sigma-1) - x \log_\sigma(x) - (1-x) \log_\sigma(1-x)$. ◀

Boyar et al. [7] investigated a problem called *maximum asymmetric string guessing* (MAXASGK). The input is a sequence of requests (x_0, x_1, \dots, x_n) where $x_0 = \perp$ and for $i \geq 1$, $x_i \in \{0, 1\}$. The algorithm has to produce a sequence of answers $(y_1, \dots, y_n, y_{n+1})$. The output is feasible if $x_i \leq y_i$ for all $1 \leq i \leq n$. The profit of the algorithm is the number of zeroes in y_1, \dots, y_n for feasible outputs, and $-\infty$ otherwise. The “blind” version of the problem, where the algorithm has to produce the outputs without actually seeing the requests (i.e., in each step, the algorithm receives some dummy request \perp), is denoted MAXASGU. In what follows, let

$$B_c := \log \left(1 + \frac{(c-1)^{c-1}}{c^c} \right) \approx \frac{1}{c} \cdot \frac{1}{e \ln 2}.$$

► **Theorem 6** (Boyar et al. [7]). *For any function $c(n)$ such that $1 \leq c(n) \leq n$, there is a c -competitive algorithm for MAXASGK (MAXASGU, respectively) with advice of size $B_c \cdot n + O(\log n)$. Moreover, any c -competitive algorithm for MAXASGK (MAXASGU, respectively) must read at least $B_c \cdot n - O(\log n)$ bits of advice.* ◀

Note that in general, it does not make much difference if the length of the input is initially known to the algorithm or not. More specifically, it changes the advice complexity by at most $O(\log n)$.

3 Max- π and Min- π without Preemption

First, we show that for any non-trivial hereditary property π , the MAX- π problem is equivalent to the asymmetric string guessing in the following sense.

► **Theorem 7.** *If there is a c -competitive algorithm for MAXASGU then there is a c -competitive algorithm for MAX- π using the same advice.*

Proof. Suppose there is a c -competitive algorithm ALG for MAXASGU with $c(n) < \infty$. For any input for MAX- π , consider a binary string representing this instance (with one bit per vertex) such that zeroes correspond to vertices in the optimal solution. Any feasible solution to MAXASGU must cover all ones in the input string, so the set of zeroes in any feasible solution of MAXASGU forms subset of vertices of the optimal solution of MAX- π , and since π is hereditary, is a feasible solution of MAX- π .

The algorithm for MAX- π works as follows: When a vertex arrives, send a \perp request to ALG. If ALG answers 0, include the vertex in the solution. Since ALG is c -competitive, it covers at least a $(1/c)$ -fraction of the optimum. ◀

► **Theorem 8.** *If there is a c -competitive algorithm for MAX- π that reads b bits of advice then there is a c -competitive algorithm for MAXASGK using $b + O(\log^2 n)$ bits of advice.*

Before proving Theorem 8 let us recall Lemma 3 from Bartal et al. [1].

► **Lemma 9** (Bartal et al. [1]). *Given any graph H , there exist constants n_0 and α such that for all $n > n_0$ there exists a graph G on n vertices such that any induced subgraph of G on at least $\alpha \log n$ vertices contains H as an induced subgraph.*

This is a variant of Lemma 9 from Lund and Yannakakis [15].

► **Lemma 10** (Lund and Yannakakis [15]). *Let H be a graph on k vertices. For sufficiently large N , for any graph G on N vertices and for all $\ell = \Omega(\log N)$, a pseudo-random subgraph G' of G does not, with probability $1/2$, contain a subset S of ℓ vertices that is a clique in G but H is not an induced subgraph of $G'|_S$.*

Proof of Theorem 8. According to Lemma 1, π is satisfied either by all cliques or by all independent sets. Without loss of generality, suppose the latter (otherwise, swap the edges and non-edges in the following arguments).

Consider a binary string $\nu = x_1, \dots, x_n$ (for large enough n). Let us consider the graph $G_\nu = (V, E)$ defined as follows. Let H be an arbitrary but fixed forbidden subgraph of π . Let G' be the n -vertex graph from Lemma 9 with vertices $V = \{v_1, \dots, v_n\}$. If $x_i = 0$ for some i , delete from G' all edges $\{v_i, v_j\}$ for $j > i$. In the graph G_ν thus defined, the vertices v_i for which the corresponding x_i satisfies $x_i = 0$ (denoted by $I_\nu \subseteq V$ in the sequel) form an independent set, and hence $G_\nu[I_\nu]$ has property π . On the other hand, any induced subgraph $G_\nu[S]$ with property π can contain at most $\alpha \log n$ vertices from $V \setminus I_\nu$ (otherwise it would contain the forbidden graph H as induced subgraph). Note that, with $O(\log n)$ bits of advice to encode n , the graph G_ν can be constructed from the string ν in an online manner: the base graph G' is fixed for a fixed n , and the subgraph $G_\nu[v_1, \dots, v_i]$ depends only on the values of x_1, \dots, x_{i-1} .

Now let us consider a c -competitive algorithm ALG_π for $\text{MAX-}\pi$ that uses b bits of advice. Let us describe how to derive an algorithm ALG for MAXASGK from ALG_π . For a given string $\nu = x_1, \dots, x_n$, where \perp, x_1, \dots, x_n is the input for MAXASGK , the advice for ALG consists of three parts: first, there is a self-delimited encoding of n using $O(\log n)$ bits, followed by a (self-delimited) correction string e_ν of length $O(\log^2 n)$ bits described later, and the rest is the advice for ALG_π on the input G_ν . Let S be the solution (set of vertices) returned by ALG_π on G_ν (with the proper advice). As argued before, S can contain at most $\alpha \log n$ vertices from $V \setminus I_\nu$. The indices of these vertices from $S_{\text{out}} := S \cap (V \setminus I_\nu)$ are part of the string e_ν . Apart from that, e_ν contains the indices of at most $\alpha \log n$ vertices $S_{\text{in}} \subseteq I_\nu$ such that $|(S \setminus S_{\text{out}}) \cup S_{\text{in}}| = \min\{|S|, |I_\nu|\}$.

The algorithm ALG works as follows: at the beginning, it constructs the graph G' . When a request x_i arrives, ALG sends the new vertex v_i of G_ν to ALG_π , and finds out whether $v_i \in S$. If $v_i \in S_{\text{in}}$, ALG answers 0 regardless of the answer of ALG_π . Similarly, if $v_i \in S_{\text{out}}$, ALG answers 1. Otherwise, ALG answers 0 if and only if $v_i \in S$.

First note that ALG always produces a feasible solution: if the input $x_i = 1$ then either $v_i \notin S$, and ALG returns $y_i = 1$, or else v_i is included in S_{out} . Moreover, the number of zeroes (the profit) in the output of ALG is $\min\{|S|, |I_\nu|\}$, where $|I_\nu|$ is the profit of the optimal solution. Since ALG_π is c -competitive, $|S| \geq (1/c) \cdot \text{profit}(\text{OPT}(G_\nu)) \geq (1/c) \cdot |I_\nu|$. ◀

► **Corollary 11.** *Let π be any non-trivial hereditary property. Let $A_{c,n}$ be the minimum advice needed for a c -competitive $\text{MAX-}\pi$ algorithm. Then*

$$B_c \cdot n - O(\log n) \leq A_{c,n} \leq B_c \cdot n + O(\log^2 n).$$

We have shown that the advice complexity of $\text{MAX-}\pi$ essentially does not depend on the choice of the property π . This is not the case with cohereditary properties and the problem $\text{MIN-}\pi$. On one hand, there are cohereditary properties where little advice is sufficient for optimality:

► **Theorem 12.** *If a cohereditary property π can be characterized by finitely many obligatory subgraphs, there is an optimal algorithm for $\text{MIN-}\pi$ with advice $O(\log n)$.*

Proof. Since each obligatory subgraph has constant size, $O(\log n)$ bits can be used to encode the indices of the vertices (forming the smallest obligatory subgraph) that are included in an optimal solution. ◀

On the other hand, there are properties for which the problem $\text{MIN-}\pi$ requires large advice.

► **Theorem 13** (Boyar et al. [7]). *Any c -competitive algorithm for the minimum cycle finding problem requires at least $B_c \cdot n - O(\log n)$ bits of advice.* ◀

The problem *minimum cycle finding* requires to identify a smallest possible set of vertices S such that $G[S]$ contains a cycle. Hence, it is the $\text{MIN-}\pi$ problem for the non-trivial cohereditary property “contains cycle.” An upper bound analogous to Theorem 7 follows from [7].

► **Theorem 14.** *Let π be any non-trivial cohereditary property. There is a c -competitive algorithm for $\text{MIN-}\pi$ which reads $B_c \cdot n + O(\log n)$ bits of advice.* ◀

4 Max- π with Preemption – Large Competitive Ratios

In this, and the following section, we consider the problem $\text{MAX-}\pi$ with preemption where π is a non-trivial hereditary property. In every time step, the algorithm can either accept or reject the currently given vertex and preempt any number of vertices that it accepted in previous time steps. However, vertices that were once rejected or preempted cannot be accepted in later time steps. The goal is to accept as many vertices as possible. After each request, the solution is required to have the property π .² Using a string guessing reduction, we prove the following theorem.

► **Theorem 15.** *Consider the $\text{MAX-}\pi$ problem with preemption, for a hereditary property π with a forbidden subgraph H , such that π holds for all independent sets. Let $c(n)$ be an increasing function such that $c(n) \log c(n) = o(\sqrt{n/\log n})$. Any $c(n)$ -competitive $\text{MAX-}\pi$ algorithm uses at least $\Omega\left(\frac{n}{c(n)^2 \log c(n)}\right)$ bits of advice.*

Proof. First, for some given n and σ , let us define the graph $G_{n,\sigma}$ that will be used in the reduction. To ease the presentation, assume that $n' = n/\sigma$ is integer. Let G_1 be a graph with σ vertices, the existence of which is asserted by Lemma 9, such that any subgraph of G_1 with at least $\kappa_1 \log \sigma$ vertices contains H as induced subgraph. Let G_B be the complement of a union of n' cliques of size σ (i.e., G_B consists of n' independent sets $V_1, \dots, V_{n'}$ of size σ , and all remaining pairs of vertices are connected by edges). Applying Lemma 10 to G_B proves an existence of a graph $G_2 \subseteq G_B$ such that any subset of G_2 with at least $\kappa_2 \log n$ vertices contains H as an induced subgraph. The graph $G_{n,\sigma}$ is obtained from G_2 by replacing each independent set V_i with a copy of G_1 (each such copy is called a “layer” in what follows).

Let us suppose that a $c(n)$ -competitive $\text{MAX-}\pi$ algorithm ALG is given that uses $S(n)$ advice bits on instances of size n . Now fix an arbitrary n , and choose $\sigma := 4c\kappa_1 \log(4c\kappa_1)$. We show how to solve instances of σ -SGKH of length $n' - 1$ using ALG . Let $q_1, \dots, q_{n'-1}$ be the instance of σ -SGKH, where $q_i \in \{1, \dots, \sigma\}$. The corresponding instance G for the $\text{MAX-}\pi$ problem is as follows: take the graph $G_{n,\sigma}$, and denote by $v_{i,1}, \dots, v_{i,\sigma}$ the vertices of the set V_i . Let v_{i,q_i} be the distinguished vertex in set V_i . Delete from $G_{n,\sigma}$ all edges of the form $\{v_{i,q_i}, v_{i',q_{i'}}\}$ where $i' > i$. The resulting graph G is presented to ALG in the order $v_{1,1}, \dots, v_{1,\sigma}, v_{2,1}, \dots, v_{2,\sigma}, \dots$

Note that G can be constructed online based on the instance $q_1, \dots, q_{n'-1}$.

² Note that without advice, the condition to maintain π in every time step is implicit. Indeed, if π should be violated in any step, the adversary may just end the input sequence. Then again, allowing advice changes this game. Here, an online algorithm may just accept all vertices, compute the optimal solution at the end, and preempt the corresponding vertices. Therefore, we require explicitly that π must be maintained in every time step.

The distinguished vertices form an independent set of size n' , and thus a feasible solution. On the other hand, apart from the distinguished vertices, any solution can have at most $\kappa_1 \log \sigma$ vertices in one layer (otherwise there would be a forbidden subgraph in that layer), and at most $\kappa_2 \log n$ layers with vertices other than the distinguished one (if there are more than $\kappa_2 \log n$ nonempty layers, choose one vertex from each nonempty layer; these form a clique in G_B , and due to Lemma 10 induce H in G_2 , and thus also in G). Hence, $n' \leq \text{profit}(\text{OPT}(G)) \leq n' + K$, where $K := \kappa_1 \kappa_2 \log \sigma \log n$.

Since ALG is c -competitive, it produces a solution of size at least $\text{profit}(\text{OPT}(G))/c$. Since any solution can have at most K non-distinguished vertices, the solution of ALG contains at least $g := \text{profit}(\text{OPT}(G))/c - K$ distinguished vertices.

Consider an algorithm ALG' for σ -SGKH of length $n' - 1$, which simulates ALG: for the i th request, it presents ALG with the layer of vertices V_i . Let $\text{Cand}(i) \subseteq V_i$ (the *candidate set*) be the set of vertices selected by ALG from V_i . As stated before, $|\text{Cand}(i)| \leq \kappa_1 \log \sigma$. A set $\text{Cand}(i)$ is *good* if it contains the distinguished vertex v_{i,q_i} . It follows from the definition of the problem that there are at least g good candidate sets.

ALG' uses an additional $O(\log \log \sigma)$ bits of advice to describe a number j , $1 \leq j \leq \kappa_1 \log \sigma$, and selects the j th vertex from any $\text{Cand}(i)$ as an answer (if $|\text{Cand}(i)|$ is smaller than j , it is extended in an arbitrary fixed way). The number j is selected in such a way that ALG' gives the correct answer from a fraction of $1/(\kappa_1 \log \sigma)$ of good sets. Putting it together, the fraction of correctly guessed numbers by ALG' is at least

$$\alpha := \frac{n' - cK}{c\kappa_1 \log \sigma (n' - 1)}.$$

Note that

$$\frac{1}{c\kappa_1 \log \sigma} \geq \alpha \geq \frac{1}{2c\kappa_1 \log \sigma}$$

holds for large enough n , provided that $n' \geq 2cK - 1$. To see that this inequality holds, note that

$$n' \geq 2cK - 1 \iff \frac{n}{4c\kappa_1 \log(4c\kappa_1)} \geq 2cK - 1 \iff (2cK - 1)4c\kappa_1 \log(4c\kappa_1) \leq n.$$

The last inequality holds for large enough n by the choice of $c(\cdot)$ due to the fact that

$$(2cK - 1)4c\kappa_1 \log(4c\kappa_1) = O(c(n)^2 K \log c(n)) = O((c(n) \log c(n))^2 \log n) = o(n).$$

Due to Theorem 4, any algorithm for σ -SGKH that correctly guesses a fraction of α numbers (for $\frac{1}{\sigma} \leq \alpha \leq 1$) on an input of length $n' - 1$ requires at least $S := F(\sigma, \alpha) \cdot (n' - 1) \cdot \log_2 \sigma$ bits of advice where

$$F(\sigma, \alpha) := 1 + (1 - \alpha) \log_\sigma \left(\frac{1 - \alpha}{\sigma - 1} \right) + \alpha \log_\sigma \alpha.$$

First, let us verify that $1/\sigma \leq \alpha$:

$$\begin{aligned} \frac{1}{\sigma} &\leq \frac{1}{2c\kappa_1 \log \sigma} \leq \alpha \\ \iff \frac{\sigma}{\log \sigma} &\geq 2c\kappa_1 \\ \iff \frac{4c\kappa_1 \log(4c\kappa_1)}{\log(4c\kappa_1) + \log \log(4c\kappa_1)} &\geq 2c\kappa_1 \\ \iff 2 \log(4c\kappa_1) &\geq \log(4c\kappa_1) + \log \log(4c\kappa_1) \\ \iff \log(4c\kappa_1) &\geq \log \log(4c\kappa_1). \end{aligned}$$

Next, we use the bounds on α to get

$$\begin{aligned}
F(\sigma, \alpha) &\geq 1 + \left(1 - \frac{1}{c\kappa_1 \log \sigma}\right) \cdot \frac{\log\left(\frac{1 - \frac{1}{c\kappa_1 \log \sigma}}{\sigma - 1}\right)}{\log \sigma} + \frac{1}{2c\kappa_1 \log \sigma} \cdot \frac{\log\left(\frac{1}{2c\kappa_1 \log \sigma}\right)}{\log \sigma} \\
&\geq 1 - \left(1 - \frac{1}{c\kappa_1 \log \sigma}\right) \cdot \frac{\log(\sigma - 1)}{\log \sigma} - \frac{1}{\log \sigma} \left[\frac{\log\left(\frac{c\kappa_1 \log \sigma}{c\kappa_1 \log \sigma - 1}\right)}{\frac{c\kappa_1 \log \sigma}{c\kappa_1 \log \sigma - 1}} + \frac{\log(2c\kappa_1 \log \sigma)}{2c\kappa_1 \log \sigma} \right] \\
&\geq \frac{1}{c\kappa_1 \log \sigma} - \frac{1}{\log \sigma} \left[\frac{1}{\ln 2(c\kappa_1 \log \sigma - 1)} + \frac{\log(2c\kappa_1 \log \sigma)}{2c\kappa_1 \log \sigma} \right]
\end{aligned}$$

and hence

$$F(\sigma, \alpha) \log \sigma \geq \frac{1}{c\kappa_1} - \frac{1}{\ln 2(c\kappa_1 \log \sigma - 1)} - \frac{\log(2c\kappa_1 \log \sigma)}{2c\kappa_1 \log \sigma}.$$

Since $\sigma \approx c \log c$, it holds

$$\frac{1}{\ln 2(c\kappa_1 \log \sigma - 1)} = o\left(\frac{1}{c}\right)$$

and for $n \mapsto \infty$

$$\frac{\log(2c\kappa_1 \log \sigma)}{2c\kappa_1 \log \sigma} \mapsto \frac{1}{2c\kappa_1}$$

yielding $F(\sigma, \alpha) \log \sigma = \Omega(1/c)$. Finally, the theorem follows by noting that $n' - 1 = \Omega(n/(c \log c))$. \blacktriangleleft

Using a similar approach, we can get a stronger bound for the independent set problem. Due to space constraints, the proof is moved to Appendix A.

► **Theorem 16.** *Let $c(n)$ be any function such that*

$$8 \leq c(n) \leq \frac{1 + \sqrt{1 + 4n}}{4}.$$

Any $c(n)$ -competitive independent set algorithm that can use preemption requires at least

$$A_{c,n} \geq \frac{0.01 \cdot \log(2c)}{2c^2} (n - 2c)$$

advice bits. \blacktriangleleft

5 Max- π with Preemption – Small Competitive Ratios

In this section, we use Theorem 5 to give bounds on small constant values of the competitive ratio for algorithms for MAX- π complementing the bounds from Theorem 15. In what follows, π is a non-trivial hereditary property and k is the size of a smallest forbidden subgraph according to π .

► **Theorem 17.** *If there is a c -competitive algorithm for MAX- π with preemption that reads $b(nk)$ bits of advice for inputs of length nk , then there exists a c -competitive algorithm for Anti- k -SGKH, which, for inputs of length n , reads $b(kn) + O(\log^2 n)$ bits of advice.*

Proof. According to Lemma 1 π is satisfied either by all cliques or by all independent sets. We assume in the following, that π is satisfied by all independent sets. We describe how to transform an instance of Anti- k -SGKH into an instance of MAX- π with preemption. The length of the instance for MAX- π with preemption will be k times as long as the length n of the Anti- k -SGKH instance. We proceed to show that a c -competitive algorithm for the latter implies a c -competitive algorithm for the former which reads at most $O((\log n)^2)$ additional advice bits.

Let $\nu = x_1, x_2, \dots, x_n$ (with $x_i \in \{1, \dots, k\}$) be an instance of Anti- k -SGKH. Consider the n -vertex graph $\tilde{G} = (V(\tilde{G}), E(\tilde{G}))$, given by Lemma 9 for a size- k smallest minimal forbidden subgraph $H = (V(H), E(H))$ from π . We recall that any induced subgraph of \tilde{G} with at least $\alpha \log n$ vertices contains H as an induced subgraph. Let us denote $V(\tilde{G}) = \{\tilde{v}_1, \dots, \tilde{v}_n\}$ and $V(H) = \{h_1, \dots, h_k\}$. We now describe the construction of a graph $G_\nu = (V(G_\nu), E(G_\nu))$, which will be the input for the given algorithm for MAX- π . To this end, let

$$V(G_\nu) = \bigcup_{i=1}^n \bigcup_{j=1}^k v_j^i,$$

$$E(G_\nu) = \{\{v_j^i, v_{j'}^i\} \mid \{h_j, h_{j'}\} \in E(H)\} \cup \{\{v_j^i, v_{j'}^{i'}\} \mid i < i', \{\tilde{v}_j, \tilde{v}_{j'}\} \in E(\tilde{G}), j \neq x_i\},$$

where we assume an ordering $v_1^1, \dots, v_k^1, v_1^2, \dots, v_k^2, \dots, v_1^n, \dots, v_k^n$ on the vertices. Moreover, we denote the requests v_1^i, \dots, v_k^i as layer i . Let X denote the set of vertices $v_{x_i}^i$ for $i = 1, \dots, n$.

We start with a few observations about G_ν that are straightforward.

- **Observation 1.** The graph $G_\nu[X]$ is an independent set of size n . In particular, it has property π .
- **Observation 2.** $G_\nu[v_1^i, \dots, v_k^i] = H$ for an arbitrary but fixed i . Thus, any induced subgraph of G_ν that contains $G_\nu[v_1^i, \dots, v_k^i]$ does not have property π .
- **Observation 3.** Consider a set of vertices, V , in G_ν which is disjoint from X . If $|V| \geq k\alpha \log n$, then $G_\nu[V]$ does not have property π .

Note that V must in this case contain vertices from at least $\alpha \log n$ different layer. These have H as an induced subgraph since none of them are in X .

Now consider a c -competitive algorithm ALG_π for MAX- π with preemption reading $b(nk)$ bits of advice (recall that nk is the length of its input G_ν). We start by describing an algorithm ALG' for Anti- k -SGKH, which uses $b(nk)$ bits of advice (n is the length of its input). Afterwards, we use it to define another algorithm ALG for Anti- k -SGKH, which uses $O(\log^2 n)$ additional advice bits and is c -competitive.

For a given string $\nu = x_1, \dots, x_n$, let \perp, x_1, \dots, x_n be the input for Anti- k -SG. Let S be the solution (set of vertices) returned by ALG_π on G_ν (with the proper advice). Note that this is the resulting set of vertices after the unwanted vertices have been preempted. ALG' works as follows: It constructs the graph G_ν online and simulates ALG_π on it. When a request i arrives, the goal of ALG' is to guess a number in $\{1, \dots, k\}$ different from x_i . It does this by presenting all vertices in layer i to ALG_π . It is important to note that the vertices in layer i can be presented without knowledge of x_i, \dots, x_n . Let S_i denote the set of these vertices, which are accepted by ALG_π and have not been preempted after request v_k^i . In layer i , ALG' outputs $y_i = w$ where w is the smallest number in $\{1, \dots, k\}$ such that $v_w^i \notin S_i$. Note that such a number always exists because of Observation 2.

We now describe ALG , which uses $O(\log^2 n)$ additional advice bits. The advice for ALG consists of three parts: First, a self-delimiting encoding of n (this requires $O(\log n)$ bits).

This is followed by a list of up to $k\alpha \log n$ indices i , where ALG outputs $y_i = x_i$. Let S_{error} denote the set of these indices. A self-delimiting encoding of this requires $O(\log^2 n)$ bits (recall that α and k are constant). Finally, the advice which ALG' received is included. This is $b(nk)$ bits.

The algorithm ALG works as follows: For each request, it does the following: If the request is not in S_{error} it outputs the same as ALG. If the request is in S_{error} it outputs another number in $[k]$.

We now argue that ALG is c -competitive. We note that the optimal offline solution to $\text{MAX-}\pi$ on G_ν contains at most $k\alpha \log n$ vertices not in X . The same of course holds for the solution produced by ALG_π . It holds that if in layer i , ALG_π accepts a vertex in X , then ALG' outputs $y_i \neq x_i$. This means that the score of ALG_π is at most $k\alpha \log n$ more than the score of ALG' . Since the score of ALG is $k\alpha \log n$ more than the score of ALG' , we have that ALG is c -competitive. \blacktriangleleft

Combining Theorems 5 and 17, we get the following corollary.

► **Corollary 18.** *For a non-trivial hereditary property, π , with a minimal forbidden subgraph of size k , the following holds: Let $1 < c < \frac{k}{k-1}$. A c -competitive algorithm for $\text{MAX-}\pi$ with preemption must read at least*

$$b \geq (1 - h_k(1/c))n \frac{\log k}{k} - O(\log^2 n)$$

bits of advice, where n is the input length. Here, h_k is the k -ary entropy function given by $h_k(x) = x \log_k(k-1) - x \log_k(x) - (1-x) \log_k(1-x)$.

6 Closing Remarks

In Corollary 11, we describe lower and upper bounds for the advice complexity of all online hereditary graph problems, which are essentially tight (there is just a gap of $O(\log^2 n)$). It turns out that, for all of them, roughly the same amount of information about the future is required to achieve a certain competitive ratio.

Intriguingly, we see a quite different picture for cohereditary properties. Theorem 14 gives the same upper bound as we had for hereditary properties, and Theorem 13 shows that this upper bound is essentially tight. However, Theorem 12 shows that there exist cohereditary problems that have an advice complexity as low as $O(\log n)$ bits to be optimal. It remains open if it is only those problems with a finite set of obligatory graphs that have this very low advice complexity, or if this can also happen for cohereditary problems with an infinite set of obligatory graphs.

For hereditary problems with preemption, we show that to achieve a competitive ratio strictly smaller than $\frac{k}{k-1}$, a linear number of advice bits is needed. This is asymptotically tight, since optimality (even without preemption) can be achieved with n bits. Furthermore, we show a lower bound for non-constant competitive ratios (that are roughly smaller than \sqrt{n}). It remains open if there is an algorithm for the preemptive case, which uses fewer advice bits than the algorithms solving the same problem in the non-preemptive case.

References

- 1 Y. Bartal, A. Fiat, and S. Leonardi. Lower bounds for on-line graph problems with application to on-line circuit and optical routing. *SIAM Journal on Computing*, 36(2):354–393, 2006.

- 2 H.-J. Böckenhauer, J. Hromkovič, D. Komm, S. Krug, J. Smula, and A. Sprock. The string guessing problem as a method to prove lower bounds on the advice complexity. *Theoretical Computer Science* 554:95–108, 2014.
- 3 H.-J. Böckenhauer, D. Komm, R. Kráľovič, and R. Kráľovič. On the advice complexity of the k -server problem. In L. Aceto, M. Henzinger, and J. Sgall, editors, *Proc. of ICALP 2011*, volume 6755 of *LNCS*, pp. 207–218. Springer-Verlag, 2011.
- 4 H.-J. Böckenhauer, D. Komm, R. Kráľovič, R. Kráľovič, and T. Mömke. On the advice complexity of online problems. In Y. Dong, D.-Z. Du, and O. H. Ibarra, editors, *Proc. of ISAAC 2009*, volume 5878 of *LNCS*, pp. 331–340. Springer-Verlag, 2009.
- 5 A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
- 6 A. Borodin and R. El-Yaniv. On randomization in on-line computation. *Information and Computation*, 150(2):244 – 267, 1999.
- 7 J. Boyar, L. M. Favrholdt, C. Kudahl, and J. W. Mikkelsen. Advice Complexity for a Class of Online Problems. In E. W. Mayr and N. Ollinger, editors, *Proc. of STACS 2015*, volume 30 of *LIPIcs*, pp. 116–129, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- 8 S. Dobrev, R. Kráľovič, and D. Pardubská. Measuring the problem-relevant information in input. *RAIRO Theoretical Informatics and Applications*, 43(3):585–613, 2009.
- 9 S. Dobrev, R. Kráľovič, Richard Kráľovič. Advice complexity of maximum independent set in sparse and bipartite graphs. *Theory of Computing Systems* 56(1):197–219, 2015.
- 10 Y. Emek, P. Fraigniaud, A. Korman, and A. Rosén. Online computation with advice. *Theoretical Computer Science*, 412(24):2642–2656, 2011.
- 11 J. Hromkovič, R. Kráľovič, and R. Kráľovič. Information complexity of online problems. In F. Murlak and P. Sankowski, editors, *Proc. of MFCS 2010*, volume 6281 of *LNCS*, pp. 24–36. Springer-Verlag, 2010.
- 12 D. Komm and R. Kráľovič. Advice complexity and barely random algorithms. *RAIRO Theoretical Informatics and Applications*, 45(2):249–267, 2011.
- 13 I. Csiszár. The method of types. *Information Theory, IEEE Transactions on* 44(6):2505–2523, 1998.
- 14 J. M. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219 – 230, 1980.
- 15 C. Lund and M. Yannakakis. The approximation of maximum subgraph problems. In A. Lingas, R. Karlsson, and S. Carlsson, editors, *Proc. of ICALP 1993*, volume 700 of *LNCS*, pp. 40–51. Springer-Verlag, 1993.
- 16 F. P. Ramsey. On a problem in formal logic. *Proc. London Mathematical Society* (3), 30:264–286, 1930.
- 17 D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.
- 18 A. C.-C. Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *Proc. of FOCS 1977*, pp. 222–227. IEEE Computer Society, 1977.
- 19 J. W. Mikkelsen. Randomization can be as helpful as a glimpse of the future in online computation. *CoRR*, abs/1511.05886, 2015.

A

 Proof of Theorem 16

Consider an instance of σ -SGKH of length $n' - 1$. We construct an unweighted undirected graph G as input for the maximum independent set problem with preemption as follows. We subdivide the input into $n' = n/\sigma$ layers. In every layer, vertex after vertex, σ vertices are given that form a clique. These cliques are denoted by $C_1, \dots, C_{n'}$; the clique C_i , $1 \leq i \leq n' - 1$, contains a designated vertex c_i such that all vertices that are revealed subsequently are connected to all vertices $v \in C_i$ if and only if $v \neq c_i$. This way, G contains an independent set (namely $\{c_1, \dots, c_{n'-1}, v\}$ for any $v \in C_{n'}$) of size n' .

Note that due to the condition that an independent set is maintained in every time step, no online algorithm is allowed to accept more than one vertex in any layer. If, for clique C_i with $1 \leq i \leq n' - 1$, the algorithm accepts a vertex $v \neq c_i$, in the next layer it has two choices. It can preempt v which reduces its profit by 1, or it can keep v and not have any more profit on the rest of the instance. Clearly, the second option is never superior to the first one, so we can assume, without loss of generality, that the algorithm always preempts incorrectly guessed vertices. Also, assuming that the algorithm always selects exactly one vertex means no loss of generality (it has no benefit not to select any vertex in a given layer).

Given a $c(n)$ -competitive independent set algorithm ALG, one can construct a σ -SGKH algorithm ALG' as follows: for each request, ALG' presents ALG with a new set of σ vertices, and the vertex selected by ALG is interpreted as the answer to σ -SGKH. The next request of σ -SGKH reveals the correct guess, so the instance for ALG can be constructed incrementally. The choice of the designated vertex c_i for C_i with $1 \leq i \leq n' - 1$ by ALG increases the profit of ALG by 1, and corresponds to the correct guess of the natural number from 1 to σ by ALG' . A wrong choice gets preempted, and does not contribute to the profit of ALG. Let

$$\alpha := \frac{n' - c(n)}{c(n)n' - c(n)}. \quad (3)$$

Since ALG is $c(n)$ -competitive, and the optimum has size n' , ALG selects an independent set of size at least $n'/c(n) = \alpha(n' - 1) + 1$, meaning that the fraction of correct guesses was at least α . On the other hand, due to Theorem 4, any algorithm for σ -SGKH that correctly guesses a fraction of α numbers (for $\frac{1}{\sigma} \leq \alpha \leq 1$) on an input of length $n' - 1$ requires at least $S := F(\sigma, \alpha) \cdot (n' - 1) \cdot \log_2 \sigma$ bits of advice where

$$F(\sigma, \alpha) := 1 + (1 - \alpha) \log_\sigma \left(\frac{1 - \alpha}{\sigma - 1} \right) + \alpha \log_\sigma \alpha.$$

We have

$$\frac{1}{c} \geq \alpha = \frac{n' - c}{cn' - c} \geq \frac{1}{2c}. \quad (4)$$

provided that $1 \leq c < \frac{n'+1}{2}$. Using (4), we get

$$\begin{aligned} F(\sigma, \alpha) &\geq 1 + \frac{c-1}{c} \cdot \frac{\log\left(\frac{c-1}{c(\sigma-1)}\right)}{\log \sigma} + \frac{1}{2c} \cdot \frac{\log\left(\frac{1}{2c}\right)}{\log \sigma} \\ &= 1 - \frac{c-1}{c} \cdot \frac{\log(\sigma-1)}{\log \sigma} - \frac{1}{\log \sigma} \cdot \left[\frac{\log\left(\frac{c}{c-1}\right)}{\frac{c}{c-1}} + \frac{\log(2c)}{2c} \right] \\ &\geq \frac{1}{c} - \frac{1}{\log \sigma} \cdot \left[\frac{\log\left(\frac{c}{c-1}\right)}{\frac{c}{c-1}} + \frac{\log(2c)}{2c} \right]. \end{aligned}$$

Since for any $x \geq 0$ it holds³ $\ln(1+x) \leq x(1+x)$, we get

$$F(\sigma, \alpha) \geq \frac{1}{c} - \frac{1}{\log \sigma} \cdot \left[\frac{1}{\ln 2(c-1)} + \frac{\log(2c)}{2c} \right] \geq \frac{1}{c} - \frac{1}{\log \sigma} \cdot \frac{\log(4(2c)^{\ln 2})}{2 \ln 2(c-1)}.$$

Subsequently, for the required advice we get

$$S \geq \frac{n-\sigma}{\sigma} \cdot \log \sigma \cdot F(\sigma, \alpha) \geq \frac{n-\sigma}{\sigma} \cdot \left[\frac{\log \sigma}{c} - \frac{\log(4(2c)^{\ln 2})}{2 \ln 2(c-1)} \right]. \quad (5)$$

We have to choose σ in such a way that $\alpha \geq \frac{1}{\sigma}$ and $c < \frac{\frac{n}{\sigma}+1}{2}$. It is easy to see that for $\sigma = 2c$ both inequalities hold for the assumed values of c ; the second inequality being equivalent to $4c^2 - 2c - n < 0$. The function

$$h(c) := \frac{\log(4(2c)^{\ln 2})}{2 \ln 2(c-1)} = \frac{2 + \ln 2 \log(2c)}{2 \ln 2(c-1)}$$

converges to $\frac{1}{2} \cdot \frac{\log(2c)}{c}$. Hence, for sufficiently large c , $h(c)$ comprises a linear fraction of $\frac{\log \sigma}{c}$. Solving numerically, for $c \geq 8$, $h(c) \leq 0.99 \cdot \frac{\log(2c)}{c}$, yielding

$$S \geq 0.01 \cdot \frac{n-2c}{2c} \cdot \frac{\log(2c)}{c},$$

which finishes the proof.

³ Let $f(x) = \ln(1+x)$, $g(x) = x(1+x)$. It holds $f(0) = g(0) = 0$. The derivatives are $f'(x) = \frac{1}{1+x}$, $g'(x) = 1+2x$. Both $f(x)$, $g(x)$ are increasing, $f'(0) = g'(0) = 1$, and $f'(x)$ is decreasing while $g'(x)$ is increasing.